SC0361WI

METHOD AND APPARATUS FOR FILLING LINES IN A CACHE

Field of the Invention

This invention relates to cache memories, and more particularly to filling lines in a cache memory.

Background of the Invention

Caches are frequently used in processing systems to improve the performance of the processing system. The cache is for providing very fast data in response to requests by the processor. Performance improves as the cache has more data that the processor wants. When it does have what the processor wants this is commonly called a hit. When it does not, it is commonly called a miss. Thus, a high hit rate is desirable. The better hit rate is achieved by having the cache contain the data that the processor is currently seeking. The processor sends out an address wanting the data at that address. The processor then sends out another address. The most likely address for the next address is one that is consecutive with the preceding one. Thus, one technique that has developed is to fetch not just the data at the requested address in the case of a miss but also data at consecutive addresses, which is called prefetching. The cache is filled with the data at the requested addresses, then the data is supplied to the processor. This is adequate if only a few additional addresses are prefetched.

Thus, there is a need for a technique to handle prefetches for a cache in a manner that more efficiently handles the prefetching for populating the cache and servicing the processor.

Brief Description of the Drawings

FIG. 1 is a block diagram of processing system according to an embodiment of the invention;

FIGs. 2 and 3 are a diagram of a portion of a line in a cache of FIG. 1; and

FIG. 4 is a timing diagram useful in understanding the circuit of FIG. 1.

Skilled artisans appreciate that elements in the figures are illustrated for simplicity and clarity and have not necessarily been drawn to scale. For example, the dimensions of some of the elements in the figures may be exaggerated relative to other elements to help improve the understanding of the embodiments of the present invention.

Description of the Invention

Described herein is an embodiment which provides a way to populate a cache line in a cache while providing needed service to a processor. The cache is populated with data from a prefetch. The prefetch is performed as a result of a miss in the cache. The processor is provided with requested data as soon as it has arrived. If during a prefetch, the processor generates a request that is a miss and is not part of the prefetch, the prefetching terminated. If during the prefetch, the processor generates a request that is a miss but is for one of the prefetched addresses, the prefetching continues.

Shown in FIG. 1 is a processing system 10 comprising an external memory interface 12, a direct memory access circuit (DMA) 14, a system interface 16, a DSP core 18, a data bus interface 20, a fetch unit 22, a cache 24,

a level 1 memory 26, a DSP core 28, a memory interface 30, and a level 2 memory 32. DSP core 18 operates with level 1 memory 26 via a bus having a program bus, a data bus 1, and a data bus 2. DSP core 18 operates with data bus interface 20 via data bus 1 and data bus 2 and with fetch unit 22 and cache 24 via the program bus. Data bus interface 20, fetch unit 22, cache 24, system interface 16, and memory interface 30 are coupled to a local bus. External memory interface 12, system interface 16, DMA 14, and Level 2 memory 32 are coupled to a system bus. Memory interface 30 and level 2 memory 32 are coupled to each other.

DSP core 18 initiates a request for data by providing an address on the program bus. If the level 1 memory 26 does not have that address, cache 24 determines if there is hit. If there is, cache 24 provides the data. If there is a miss, fetch unit 22 initiates a request for the data and provides the address to system interface 16 and level 2 memory 32 via memory interface 30. If the address is not for level 2 memory 32, system interface 16 passes the address on to external memory interface 12, which in turn passes it on to an external memory (not shown). This address for the data that was missed in the cache is herein called a fetch address. After this fetch address has been sent, fetch unit also initiates additional requests for data at address locations that are consecutive with the fetch address. These addresses are not actually requested at this point in time, so these are called prefetch addresses. The fetch address and the prefetch addresses are all in a single line in the cache and are consecutive addresses. In this particular example, cache 24 has 16 words in a line. A word can be any length and in this case is 128 bits. Thus each fetch and prefetch is for 128 bits of data.

When the data at the fetch address is received from external memory by external memory interface 12, the data is coupled to cache 24 via system interface 16. Similarly, if level 2 memory 32 has the fetch address, level 2 memory provides the data to cache 24 via memory interface 30. In either case, the data is provided onto the local bus where it is simultaneously provided to DSP core 18 on the program bus via fetch unit 22 and loaded into cache 24. With regard to the prefetch addresses, the data is similarly returned to cache 24 via the local bus. Fetch unit 22 sends out prefetch addresses as rapidly as it is possible to do so. At least one constraint on this can include how many pending addresses external memory can receive. The fetch unit tracks the progress by setting a pointer for the fetch address and moves it as the data arrives to keep track of what data has arrived. Similarly, a pointer is set for the prefetch addresses as well. The pointer moves as the prefetch addresses are sent out. Thus, there is pointer for where the first address is, one for the last prefetch address sent out, and one for the location where the latest data was received.

During the time at least some of the prefetch addresses are outstanding, DSP core may make another request. If it is a miss in cache 24, but is for one of the outstanding prefetch addresses, then fetch unit 22 will not attempt to change the fetching that is occurring and will provide the data as it comes in. In the alternative, if the request is a miss and is not for an outstanding prefetch address, then fetch unit 22 terminates the outstanding prefetches and initiates a request for the new request. The address for the new request then becomes the fetch address and new prefetches are initiated. There is then new pointers for the fetch address, the data as it comes in, and the last prefetch address that was sent out.

The pointer operation can be seen in FIGs. 2 and 3, which show a line 34 in cache 24. Line 34 has 16 words shown as 0-15. This shows an example in which the prefetch address was word 1. FIG. 2 shows a prefetch end pointer that moves as the prefetch addresses are sent out. The initial setting is at the fetch address and moves as the prefetch addresses are sent out. In this example, the prefetch pointer shows that the address for word 12 was the last prefetch address sent out. Shown in FIG. 3 is the data pointer, which shows that data has been received through word 4.

Shown in FIG. 4 is a timing diagram useful in understanding the sequence of operation of a particular example. P0 in FIG. 4 represents the fetch address. In this case DSP core 18 requested address P0, the fetch address, followed by prefetch addresses that are consecutive addresses P0+1, P0+2, P0+3, P0+4, P0+5, P0+6, P0+7, P0+8, P0+9, P0+10, and P0+11. These are provided on a local address bus portion of the local bus. The local bus comprises the local address bus and a local data bus. Processing system 10 allows for up to two outstanding requests by DSP core 18. In this example, request for P0+1 occurs just after data for P0 has been received as shown in the cache data. Core freeze is initiated as the request for P0+2 is initiated to indicate that no more requests can be made because there are then two outstanding requests. Core freeze is terminated after the data for P0+1 has arrived. This further shows that the P0+2 request remains pending until after the data for P0+2 has arrived. By the time the request for P0+3 arrives, the data for P0+3 has arrived so that there is then a hit in cache 24. In this case, so long as there were no further requests, the line that P0 is in would be filled from the P0.

This also points out that the priority given for a fetch address is higher than for a prefetch address. This shows that there is a four cycle delay between the data for P0 and P0+1. Whereas, if the prefetch addresses were given the same priority, there would be no cycles between them as shown in between P0+2 and P0+3. This is not an inherent delay, but it does show what can happen. Under other circumstances there may not be any intervening priority such as may arise from servicing DSP core 28, and the data for the prefetch addresses may be arrive immediately following the data for the fetch address.

The situation shown in FIG. 4 is an example of a prefetch hit. The two requests, the requests for P0+1 and P0+2 were misses in the cache, but they were for addresses that were being prefetched. Equipping the DSP core with the data as it arrives is beneficial. DSP core 18 benefited from immediately receiving the data from P0+1 and P0+2. The generation of the request for P0+3 was thus generated prior to all of the data for the prefetch addresses being received. Recognizing the prefetch hit by fetch unit 22 results in the generation signal that indicates that the data has already be requested.

For the case in which the data for the prefetch addresses, the addresses in addition to the address that came from DSP core 18 as a request for data, has not a arrived and another request, different from the prefetch addresses, then there is both a miss in cache 24 and a prefetch miss. This results in fetch unit 22 terminating the outstanding prefetches, but there may be memory types in external memory that do not allow requests for data to be terminated after they have been received. If that is the case, the automatic generation of the prefetch addresses in that sequence is terminated, but the data that is received is loaded in cache 24. This provides for a partial line fill. A line, such as line 34, is then

only filled to the extent of the prefetches until the next fetch address that is not also a prefetch address is received.

In the foregoing specification, the invention has been described with reference to specific embodiments. However, one of ordinary skill in the art appreciates that various modifications and changes can be made without departing from the scope of the present invention as set forth in the claims below. Accordingly, the specification and figures are to be regarded in an illustrative rather than a restrictive sense, and all such modifications are intended to be included within the scope of present invention.

Benefits, other advantages, and solutions to problems have been described above with regard to specific embodiments. However, the benefits, advantages, solutions to problems, and any element(s) that may cause any benefit, advantage, or solution to occur or become more pronounced are not to be construed as a critical, required, or essential feature or element of any or all the claims. As used herein, the terms "comprises," "comprising," or any other variation thereof, are intended to cover a non-exclusive inclusion, such that a process, method, article, or apparatus that comprises a list of elements does not include only those elements but may include other elements not expressly listed or inherent to such process, method, article, or apparatus.